manuale

3

Per i possessori di computer

ZX SPECTRUM e TI 99/4A

PSEUDOCODICE:
- Istruzioni read e data
- Figura richiama
- Grafico a barre

IL BASIC DELLO SPECTRUM

I LINGUAGGI DI PROGRAMMAZIONE

TI Super sound

EV EDITORIALE VIDEO

Da questo numero COMPUTING VIDEOTECA accoglie, nella sua cerchia di amici, anche i possessori di computer TI 99/4A. Allargare il raggio delle notizie e dei programmi non significa assolutamente togliere qualcosa ai lettori già affezionati, per intenderci ai possessori di ZX SPECTRUM: al contrario ci siamo sforzati di arricchire rivista e programmi e vi spieghiamo come.
Parliamo subito di fatti e vediamo quali sono gli articoli di questo mese. Lo pseudocodice, nella sua terza puntata, tratta di istruzione READ e DATA, di figura richiama e di grafico a barre. Quindi si passa al primo di una serie di articoli preziosi sul basic dello Spectrum. Ce n'è già di che accontentare i palati difficili ed esigenti ma andiamo avanti. Un articolo sui linguaggi di programmazione e un altro sull'informazione con sistemi automatizzati formano il cuore culturale del numero. Quindi si esamina come lo SPECTRUM memorizza i dati e quali siano le capacità sonore del TI 99.
I programmi contenuti in cassetta, inoltre, sono belli e sofisticati più che mai. La cassetta presenta da un lato cinque programmi per lo ZX SPECTRUM e dall'altro altrettanti programmi per il TI 99/4A. Come vedete COMPUTING VIDEOTECA ha prodotto un notevole sforzo per dare di più ai suoi lettori. L'idea di accogliere nel nostro gruppo di amici anche gli appassionati del TI è una delle tante che è stata suggerita giusto da alcuni lettori e noi l'abbiamo accolta assieme ad altri suggerimenti. È chiaro che da parte della redazione e della casa editrice si sia richiesto uno sforzo quasi doppio ma questo è un altro discorso: il successo della nostra rivista software e la simpatia dei lettori ripagano ampiamente ogni sforzo e motivano anche i sacrifici.
L'avventura di COMPUTING VIDEOTECA continua sempre più entusiasmante e le sorprese si annunciano sempre più piacevoli.
Buon divertimento a tutti.

• ISTRUZIONI READ E DATA
• FIGURA RICHIAMA
• GRAFICO A BARRE

Con questo terzo articolo della nostra serie tratteremo due argomenti nuovi: il primo è l'acquisizione dati da programma ed il secondo... beh, rimandiamolo a più tardi, in modo che sia una sorpresa.
L'acquisizione dati da programma è un problema molto grosso, e merita una trattazione appropriata.
Dunque, fino ad ora i nostri programmi lavoravano su variabili di varia struttura (semplici o ad indice), ma comunque sempre variabili il cui valore iniziale veniva impostato in un qualche punto del programma attraverso una assegnazione (per esempio A = 20, oppure B$ = "NOME").
Se ricordate, nel programma per la ricerca del massimo tra gli N valori di una lista presentato nel secondo articolo, ognuna delle N variabili veniva riempita con un valore numerico attraverso una assegnazione. Per comodità quel programma è riprodotto nella figura 1, e potete constatare che le istruzioni 10-50 sono dedicate alla assegnazione dei valori iniziali della lista A.
Di certo questo non è il modo migliore di risolvere il problema di trovare il massimo in una lista di numeri: infatti, se volessimo cambiare i valori della lista dovremmo modificare il programma, cambiando le assegnazioni. Questo può essere faticoso oltre ad esporci al rischio di sbagliare. Da ultimo, poi, c'è il fatto che, se un programma è grosso, bisogna andare a cercare tutti i punti in cui si assegnano valori iniziali alle variabili di input e modificare le assegnazioni.
Un'idea è quella di separare l'assegnazione delle variabili dal corpo del programma, e di scrivere per esempio sulle ultime righe del programma l'elenco dei valori numerici che vogliamo assegnare alle variabili. È questo il concetto che sta dietro all'istruzione DATA in BASIC. Osservate il programma della figura 2. È lo stesso programma dell'esempio 1, ma ora i dati sono messi in una lista di 20 valori, separati da una virgola e divisi su due istruzioni DATA. Se volessimo modificare i valori basterebbe modificare i valori di queste istruzioni DATA. Ma cosa permette al programma di attribuire questi valori ad ognuna delle variabili della lista A???
Osservate la ripetizione delle istruzioni 20 ÷ 40: si ripete 20 volte l'istruzione READ A(I), dove I varia da 1 a 20. Ogni volta che esegue l'istruzione READ il BASIC cattura un valore dalla lista di valori della prima istruzione DATA che trova e la assegna alla variabile A(I).
Prosegue poi, saltando la virgola, a posizionarsi sul valore successivo. Quando ha terminato

la prima riga di valori, il BASIC si posiziona sulla prossima istruzione DATA e continua a "mangiarsi" i valori che legge come se si trovassero tutti su una stessa riga.
Finiti tutti i valori, se eventualmente la READ chiedesse un'altra lettura, il BASIC ci comunicherebbe di avere finito i dati, ma noi faremo sempre in modo di leggere tutti e solo i dati che ci servono.
Se nel corso del programma ci fosse una successiva READ il BASIC andrebbe a cercare il prossimo dato, nella lista dei valori della DATA corrente, se ne contiene ancora, oppure prenderebbe il primo della successiva istruzione DATA.
L'esempio 3 ci mostra un programma che legge il numero di elementi della LISTA A e poi carica gli elementi contenuti nella successiva istruzione DATA nella LISTA A.
Osservate la flessibilità del programma; basta variare i dati contenuti nella DATA (naturalmente con coerenza!) per eseguire sempre correttamente il caricamento dei valori A(I).
Questo è il livello di flessibilità dell'istruzione DATA. Ma esiste ancora un limite alla possibilità di usare agevolmente questo programma: abbiamo ancora il compito di modificare un pezzo del programma, per quanto separato dal corpo delle altre istruzioni, per modificare i dati su cui esso lavora. Il massimo della indipendenza si ha quando dati (di INPUT) e programmi risiedono in posti diversi. Nasce così il concetto di FILE di dati: su un elemento di registrazione separato memorizziamo i dati di INPUT del programma e poi nel programma scriviamo le istruzioni di READ che permettono di leggere il file esterno. È chiaro che la READ di un file esterno dovrà anche portare l'indicazione del nome del file da leggere, oltre che il nome della variabile a cui assegnare il valore letto, e questo è ovvio per il fatto che, a questo punto, possiamo pensare di avere a disposizione contemporaneamente molti files con dati diversi.
Ora che conosciamo il file possiamo vedere la DATA come una riduzione del concetto generale di file dati di INPUT: si tratta di un file dati particolare, che si trova in memoria ed in coda alle istruzioni del programma, ed al quale si accede senza specificare nulla in una READ del programma.
Detto questo, chiudiamo il discorso sui files dati, perché sono un argomento molto complesso: da ciò che abbiamo detto però che non si scateni la fantasia del lettore per cercare di immaginarsi cosa ci sta sotto: dirò solo che per questa strada si arriva al concetto di "data base integrato" e di macchina con intelligenza superiore (ancora nella fase di ricerca).
Un modo alternativo di ottenere dati dall'esterno, invece che assegnarli direttamente alle variabili del programma è quello di richiederli attraverso l'istruzione INPUT, facendo emettere un messaggio di richiesta del dato interessato. Questo sistema si rivela efficace per piccole quantità di dati, o per instaurare un dialogo col computer.
Il programma di figura 4, risolve

il problema dell'immissione di dati con istruzioni INPUT, che in BASIC hanno l'effetto di accettare i valori introdotti dall'utente con l'uso della tastiera. Vi invito a provare il programma 4.

Riassumendo, ci sono 4 modi per inoltrare ad un programma i dati di INPUT:

— 1. assegnare direttamente i valori di INPUT alle variabili di programma
— 2. leggere le schiere di dati attraverso una o più istruzioni di READ semplici, che fanno riferimento ai valori scritti nelle relative istruzioni DATA
— 3. memorizzare i dati su un file esterno, e richiamare le varie registrazioni ("RECORD") in memoria, attraverso l'istruzione READ con l'aggiunta dell'indicazione del file da leggere
— 4. richiedere i dati attraverso domande a video direttamente all'utente del programma.

COSTRUZIONE DI UN GRAFICO A BARRE

Per passare al prossimo argomento, concentriamoci su un problema concreto: emettere una lista di dati numerici sotto forma di grafico a barre: cioè per ogni valore numerico stampare una barra di asterischi proporzionale al valore in esame.

Proviamo a stendere una prima soluzione in pseudocodice del problema:

```
INIZIO_Grafico a barre.
  RIPETI
  PER I CHE VARIA DA 1 A N
   RIPETI
    PER J DA 1 A V (I)
     Stampa 1 asterisco di
     seguito al precedente
    FINE RIPETI
    Salta a riga nuova
   FINE RIPETI
FINE.
```

In questo programma si suppone che gli N valori siano memorizzati nella lista V. Con il ripeti più esterno si considerano uno ad uno tutti gli N valori, e per ciascuno di essi, nel ciclo più interno si stampa un asterisco per ogni unità del valore interessato.

Questo piccolo algoritmo funziona bene a patto che:

1) i valori contenuti in V non siano superiori al numero di caratteri possibili in una riga di stampa,
2) che i valori non siano più piccoli dell'unità: per esempio se i valori sono tutti compresi tra zero ed 1 non compare nessun grafico.

Questo problema si chiama il problema del "dimensionamento dinamico". Si risolve applicando una semplice proporzione: supponiamo di conoscere il massimo valore contenuto in V. Allora è giusto che il massimo valore di V sia rappresentato con il massimo numero di asterischi nella nostra riga, per esempio 30. Gli altri valori saranno rappresentati da un numero di asterischi proporzionali, in modo che un numero uguale alla metà del massimo sia rappresentato da 15 asterischi, uno uguale ai 2/3 sia rappresentato da 20 asterischi e così via. La proporzione che ci sta sotto è la seguente:

$$V : X = M : 30$$

dove V è un valore tra quelli della lista, X è il numero degli asterischi che lo rappresenta, ed M è il massimo valore tra quelli di V.
Perciò la formula che esprime X in modo esplicito è la seguente:

$X = (30 \times V)/M$

Lo pseudocodice diventa ora:

```
INIZIO. Grafico a barre con
          dimensionamento
          dinamico.
   Leggi i valori di V(N).
   Trova il massimo tra gli N
   valori ed assegnalo a MAX.
   Modifica i valori di V
   moltiplicandoli per il
   fattore di scala (cioè 30/M).
   Stampa il grafico a barre.
FINE.
```

Ora, il fatto è che noi possediamo già sia il programmino che legge una lista di N valori (Esempio 3), che il programmino che calcola il massimo tra N valori (Esempio 2).
Possiamo utilizzare i programmi già pronti, mantenendone l'indipendenza, facendo uso della istruzione di pseudocodice RICHIAMA:

```
INIZIO Grafico a barre.
   RICHIAMA "Leggi i valori"
   RICHIAMA "calcola
             massimo"
   RICHIAMA "Aggiusta i
             valori"
   RIPETI
   PER I DA 1 A N
     RIPETI
     PER J DA 1 A V(I)
       Stampa 1 asterisco di
       seguito
     FINE_RIPETI
     Salta a nuova riga
   FINE_RIPETI
```

L'istruzione RICHIAMA significa: esegui il programmino indicato e poi ritorna ad eseguire l'istruzione seguente.
In BASIC la figura RICHIAMA si traduce con GOSUB, cioè "vai a SUBroutine", dove "subrotine" è una parola inglese, ormai entrata nel gergo, che significa "sottoprogramma".
Le istruzioni di una subroutine devono essere concluse dalla istruzione RETURN, che ha l'effetto di restituire il controllo al programma principale.
Ogni subroutine è identificata dal numero della sua prima istruzione, perciò l'istruzione GOSUB avrà come operando proprio questo numero.
Nell'esempio 5 si trova il testo del programma per il grafico a barre su una lista V di 10 valori. Oltre alle cose già dette sono state aggiunte le istruzioni per stampare un rudimentale asse di riferimento, e sono stati aggiunti, accanto alle barre, i valori originali della lista.
Come possibile esercizio si può provare a parametrizzare la ampiezza massima della riga, leggendola con una READ, ed anche a rendere variabile il numero di valori da mettere in grafico. Una ulteriore variante del programma potrebbe essere il mettere il grafico "in piedi": farlo ruotare cioè di 90° in senso antiorario.
Vi accorgerete che l'algoritmo cambia totalmente: conviene costruirlo mediante un vettore che rappresenta una riga del grafico. La prima riga corrisponderà ai valori con la "quota" più alta. Se ve ne sono (si scandisce la lista V), le colonne corrispondenti della riga porteranno un asterisco. Poi si diminuisce

la quota e così via.
In pseudocodice:

```
INIZIO.
  RICHIAMA "Leggi valori"
  RICHIAMA "Calcola
              Massimo"
  RICHIAMA "Aggiusta valori"
  RIPETI
  PER NQ DA 1 A 30
   QUOTA = 30—(NQ—1)
   RIPETI
   PER I DA 1 A N
    SE V(I) ⩾ QUOTA
    ALLORA
      RIGA$ (I) = '*'
    FINE_SE
   FINE_RIPETI
   Stampa RIGA$ (da 1 a N)
  FINE_RIPETI
FINE.
```

La traduzione in BASIC di questo algoritmo può esser semplice ed istruttiva.
Ancora una volta arrivederci alla prossima puntata e...buon divertimento.

M.S

```
10 DIM A(20)
20 LET A(1)=14: LET A(2)=15
30 LET A(3)=20: LET A(4)=28
40 LET A(5)=34: LET A(6)=24
50 LET A(7)=45: LET A(8)=23
60 LET A(9)=32: LET A(10)=20
70 LET A(11)=55: LET A(12)=25
80 LET A(13)=22: LET A(14)=24
90 LET A(15)=30: LET A(16)=73
130 LET A(17)=22: LET A(18)=33
110 LET A(19)=20: LET A(20)=40
120 LET C=A(1): LET IX=1
130 FOR I=1 TO 20
140 IF C<A(I) THEN  LET C=A(I): LET IX=I
150 NEXT I
170 PRINT "IL MASSIMO VALE ";C;" CHE"'"CORRISPONDE ALLA VARIABILE A(";IX;")"
```

Esempio 1. Programma che estrae il massimo valore nella lista A(N), e scrive anche il nome della variabile che lo contiene.

```
10  DIM A(20)
20  FOR I = 1 TO 20
30  READ A(I)
40  NEXT I
50  C = A(1)
60  FOR I = 1 TO 20
70     IF C<A(I) THEN C = A (I) : IX = I
```

```
90  NEXT I
100 PRINT "IL MASSIMO VALE" C "che corrisponde a A("IX")".
120 DATA 14,15,20,28,34,24,45,23,32,20
140 DATA 55,25,22,24,50,73,22,33,20,40
```

Esempio 2. Programma che estrae il massimo valore nella lista, con istruzione READ e DATA.

```
10 READ N
20 DIM A(N)
30 FOR I = 1 TO N
40 READ A(I)
50 NEXT I
60 DATA 10
70 DATA 5,7,9,11,13,15,17,19,21,23
```

Esempio 3. Programma che legge una lista di lunghezza variabile.

```
10 INPUT "QUANTI VALORI"; N
20 DIM A(N)
30 FOR I = 1 TO N
40 PRINT "DAMMI IL VALORE"; I;
50 INPUT A(I)
60 NEXT I
70 PRINT "Ecco Fatto"
```

Esempio 4. Programma che legge N valori immessi da tastiera.

```
10 DIM V(13): LET MAX=0
20 PRINT "+---------1---------2---------3"
30 PRINT "+----5----0----5----0----5----0"
40 GO SUB 185: REM LEGGE VALORI
50 GO SUB 110: REM CALCOLA MASSIMO
60 GO SUB 145: REM AGGIUSTA VALORI
70 FOR I=1 TO 10
80 LET K$=STR$ (INT V(I)): IF LEN K$<2 THEN  LET K$="0"+K$
85 PRINT K$;: PRINT "  ";
90 FOR J=1 TO V(I)
100 PRINT "*";
101 NEXT J
102 PRINT
```

```
133 NEXT I
105 STOP
110 REM ROUTINE TROVA MASSIMO
115 FOR I=1 TO 10
120 IF V(I)>MAX THEN  LET MAX=V(I)
125 NEXT I
130 RETURN
145 REM ROUTINE AGGIUSTAMENTO VALORI
150 FOR I=1 TO 10
160 LET V(I)=(30*V(I))/MAX
170 NEXT I
180 RETURN
185 REM ROUTINE LETTURA VALORI
190 FOR I=1 TO 10
200 READ V(I)
210 NEXT I
220 RETURN
1000 DATA 100,150,200,30,68,120,200,140,130,110
```

Esempio 5. Programma che costruisce il grafivo a barre.

```
+---------1---------2---------3
+----5----0----5----0----5----0
15  ***************
22  **********************
30  ******************************
04  ****
10  **********
18  ******************
30  ******************************
21  *********************
19  *******************
16  ****************
```

Esempio 6. Ecco il grafico prodotto dal programma dell'Esempio 5.

IL BASIC DELLO SPECTRUM

Sul numero scorso abbiamo visto i criteri da seguire per un corretto sviluppo dei programmi a partire dalle idee iniziali. Con questo articolo inizieremo un piccolo corso di BASIC per quanti non conoscono ancora molto bene questo diffusissimo linguaggio di programmazione. Naturalmente nel nostro caso faremo riferimento al BASIC dello Spectrum, comunque in linea generale il discorso si può applicare alla maggior parte dei computer.
Come ben saprete, il vostro computer non è in grado di capire direttamente ciò che voi volete dirgli, poiché esso comprende solo un certo numero di istruzioni elementari espresse sotto forma di numeri, le quali rappresentano appunto il linguaggio macchina del computer. Per ovviare a questo inconveniente sono stati ideati appositi interpreti che traducono le vostre istruzioni (programma BASIC) in corrispondenti serie di istruzioni in codice macchina. Naturalmente per rendere comprensibili le istruzioni, esse dovranno essere scritte secondo una certa sintassi e rispettando determinate regole. Tutto questo costituisce un linguaggio di programmazione.
Tra gli home e personal computer il linguaggio di programmazione più diffuso è sicuramente il BASIC. Ciò deriva dalla sua semplicità d'uso e dalle sue caratteristiche che lo rendono adattabile a moltissimi impieghi. Esaminiamo ora più da vicino la struttura di questo famoso linguaggio per computer.
Bisogna innanzitutto precisare che lo Spectrum adotta un sistema particolare di scrittura dei comandi BASIC, essi infatti non vengono introdotti lettera per lettera come negli altri computer, ma a ogni tasto sono associate diverse funzioni che dipendono dallo stato in cui si trova il cursore e dagli shift premuti. Benché i possessori di spectrum dovrebbero essere in grado di padroneggiare tranquillamente la tastiera di questo computer, vedremo brevemente come si ottengono le diverse istruzioni.
All'accensione del computer, all'inizio di una nuova linea, dopo un THEN o dopo i:, il cursore si trova nello stato K, premendo un qualsiasi tasto, a eccezione di quelli numerici, otterrete la parola chiave (keyword) riportata in bianco sul tasto stesso. Subito dopo avere battuto il comando, il cursore tornerà nel modo L, permettendo di scrivere le lettere minuscole. Per ottenere le istruzioni stampate in verde sopra i tasti occorre entrare nel modo E, il quale si ottiene premendo contemporaneamente CAPS SHIFT e SYMBOL SHIFT. Come in precedenza, una volta battuto il comando il cursore tornerà allo stato L. Le istruzioni in rosso poste al di sotto di ogni tasto si ottengono anch'esse nel modo E, ma in questo caso bisogna premere anche SYMBOL SHIFT insieme al tasto desiderato. Vediamo in-

fine come si accede ai simboli e alle istruzioni in rosso presenti sui tasti. Sia che siate in modo K o L, per ottenere questi simboli basterà premere SYMBOL SHIFT insieme al tasto su cui è presente il comando desiderato. Bene, ora che avete imparato come districarvi fra una miriade di shift e modi grafici (pensate che non li abbiamo nemmeno nominati tutti!), sarete senz'altro pronti per passare a qualcosa di più impegnativo.
La prima istruzione che prendiamo in esame è la nota PRINT.
Anche se tutti dovrebbero sapere a cosa serve lo ripetiamo: tramite la PRINT è possibile scrivere qualcosa sullo schermo.
La sintassi normale è:

```
PRINT "frase"
```

In questo modo verrà stampato tutto ciò che è contenuto fra gli apici. Se usate due PRINT in successione la frase contenuta nella seconda sarà stampata sotto quella contenuta nella prima. Provate:

```
PRINT "prima prova": PRINT
"seconda prova"
```

Ciò è causato dal fatto che i due punti dopo la prima istruzioni stanno a indicare che ne seguirà un'altra sulla stessa linea e quindi il computer lo interpreta come un segnale di ritorno (ENTER) andando per così dire "a capo". Se noi però mettiamo dopo la prima PRINT il ";", il computer non sposterà la posizione di stampa e quindi ciò che segue verrà stampato in successione. Per convicervi provate:

```
PRINT "prima prova"; :PRINT
"seconda prova"
```

Naturalmente dopo ogni istruzione (o serie di istruzioni) dovrete premere il tasto ENTER.
Nel caso in cui vogliate stampare dei numeri invece dei caratteri non occorre mettere gli apici.
In questo caso potrete eseguire anche operazioni matematiche all'interno della PRINT. Battete quanto segue:

```
PRINT 3 + 9-1
```

Sullo schermo apparirà il numero 11. Provate voi stessi con calcoli più complessi, ricordando però che la moltiplicazione si ottiene con il simbolo * mentre la divisione con il /.
Con questa semplice istruzione è naturalmente possibile stampare anche variabili numeriche o alfanumeriche. Vediamo prima cos'è in realtà una variabile e come si distinguono tra loro diversi tipi di variabili.
Una variabile non è altro che una o più lettere che rappresentano un numero. L'assegnazione di un valore a una variabile avviene tramite l'istruzione LET che non può essere omessa.
L'esempio seguente crea una variabile di nome PIPPO contenente il numero 125.

```
LET PIPPO = 125
```

Nel BASIC dello Spectrum la lunghezza dei nomi per le variabili numeriche non è soggetta a limitazioni benché sia consigliabile non esagerare.
Le variabili di questo genere possono contenere qualunque numero reale compreso tra circa 4 E-39 e 1 E + 38, dove E rappresenta l'esponente di 10 per cui è moltiplicato il numero, quindi 1 E + 38 equivale a scrivere 1 seguito da 38 zeri.
Oltre a queste che vengono chiamate variabili numeriche

reali, esiste un altro tipo di variabili che sono però alfanumeriche, contengono cioé lettere e simboli grafici al posto di valori numerici. Queste ultime vengono chiamate variabili stringa poiché sono appunto destinate a rappresentare stringhe di caratteri (una stringa è un qualunque insieme di simboli grafici). Le variabili stringa si differenziano dalle altre perché terminano con il simbolo $ e il loro contenuto deve essere compreso fra gli apici. Seguono alcuni esempi.

```
LET C$ = "pippo"
LET STRINGA$ = abc! # */123
+ = "
LET V$ = "120 + 250"
```

Se noi ora proviamo a stampare V$ otterremo esattamente quanto contenuto fra gli apici, cioè:

```
120 + 250
```

Questo pone in risalto la differenza esistente tra le variabili numeriche e alfanumeriche, in quanto se avessimo scritto:

```
LET V = 120 + 250
```

la variabile V conterrebbe il valore 370, ciò non è avvenuto perché inserendo il contenuto fra apici i numeri che si trovano all'interno vengono trattati come qualunque altro simbolo grafico. Per concludere l'argomento specifichiamo che sullo spectrum il contenuto di una variabile alfanumerica può essere di qualunque lunghezza si desideri. Tornando alla nostra PRINT, possiamo concludere dicendo che per stampare una variabile basta farla seguire all'istruzione stessa, si possono anche stampare testi e variabili insieme e perfino variabili stringa e numeriche in una stessa PRINT. Vediamo come:

PRINT A (stampa il numero contenuto in A)
PRINT "il numero è": A (stampa la frase fra virgolette seguita dal valore di A)
PRINT C$;A (stampa la stringa C$ seguita dal numero A)

Torneremo più avanti su questi argomenti, per adesso passiamo a un'altra istruzione fondamentale nella programmazione BASIC: INPUT.
Tramite questo comando il computer si ferma e visualizza il cursore lampeggiante in attesa che vengano inseriti dei dati; questi ultimi possono essere sia numerici che stringhe. La natura dei dati che il computer si aspetta che vengano introdotti dipende dalla variabile che segue INPUT, la quale deve essere specificata in uno dei modi permessi. Vediamo alcuni esempi:

```
INPUT A
INPUT A$
```

Questa è la forma più semplice dell'istruzione ma spesso non è sufficiente in quanto è necessario comunicare all'utente il tipo di dati richiesti. Ciò è possibile nel modo seguente:

```
INPUT "scrivi il nome";A$
```

Ricordate di non rispondere mai con delle stringhe alle richieste di INPUT numerici.
Bene, ora dovreste avere capito cosa è una variabile, come si può assegnare un valore a una di esse tramite LET o INPUT e come è possibile visualizzare sullo schermo dei dati numerici

o alfanumerici.
A questo punto possiamo introdurre il concetto di programma. Finora infatti abbiamo usato le istruzioni solo come comandi diretti i quali venivano immediatamente eseguiti premendo il tasto ENTER. Un programma è invece un insieme di molte istruzioni BASIC precedute da numeri detti appunti numeri di linea. Queste istruzioni non vengono eseguite subito, ma rimangono in memoria, pronte a essere interpretate in sequenza ogni qualvolta lo si desideri. Come abbiamo detto, nei programmi BASIC le linee sono precedute da un numero che serve per ordinarle secondo la sequenza desiderata. Ciò può risultare molto comodo poiché se si commettono delle dimenticanze è possibile aggiungere nuove linee; è anche permesso saltare indifferentemente a una qualunque linea di programma benché ciò non sia un buon metodo di programmazione. È bene limitare il più possibile le istruzioni di salto allo scopo di creare un programma omogeneo e di facile interpretazione. Vediamo comunque quali sono questi comandi e come si usano. La prima e più semplice istruzione di salto è GO TO la quale riprende l'esecuzione del programma dalla linea specificata. Se per esempio vogliamo fare un programma che continui a chiedere un numero senza mai fermarsi potremmo procedere nel modo seguente:

```
10 INPUT "numero?"; N
20 PRINT N
30 GO TO 10
```

Questo è un semplicissimo esempio di programma che continua ripetutamente a chiedervi un numero e lo stampa sullo schermo. La numerazione delle linee non deve essere necessariamente di 10 in 10 ma sarebbe buona abitudine seguire questo metodo, in modo da lasciare sufficiente spazio per l'eventuale inserimento di altre linee di programma fra quelle già esistenti.
La seconda istruzione di "salto" è la GO SUB, simile alla precedente, tranne per il fatto che questa non esegue un salto assoluto alla linea di programma specificata, ma alla subroutine che inizia a quella linea, la quale deve necessariamente terminare con un'istruzione RETURN. Ma cos'è una subroutine?
Semplice, una subroutine è una limitata (ma non necessariamente) serie di istruzioni alle quali si deve accedere da diversi punti del programma; per evitare di doverle riscrivere ogni volta è buona abitudine trasformarle in routine da inserire in coda al programma. Quando voi chiamerete la subroutine desiderata essa verrà eseguita normalmente finché non incontrerà l'istruzione RETURN, a quel punto l'esecuzione del programma riprenderà dalla linea successiva a quella di chiamata.
Per chi non avesse ancora le idee molto chiare, alla fine di questo articolo riportiamo alcuni semplici listati che riassumono gli argomenti precedentemente esposti.
Cercate di assimilare bene questi semplici temi perché nel prossimo numero passeremo a qualcosa di decisamente più interessante. Non perdetevelo!
1 - continua

Massimo Cellini

```
10 INPUT "SCRIVI IL TUO NOVE ";A$
20 PRINT "IL TUO NOME E ";A$
30 GO TO 10
```

1. Programma che richiede un nome e lo stampa sullo schermo.

```
10 INPUT "SCRIVI LA MISURA IN POLLICI ";P
20 LET C = P*2.54
30 PRINT P;" POLLICI SONO ";C; " CENTIMETRI"
40 GO TO 10
```

2. Programma di conversione da pollici a centimetri.

```
10 INPUT "BATTI IL TUO ANNO DI NASCITA ";A
20 INPUT "BATTI IL MESE ";M
30 INPUT "BATTI L'ANNO ATTUALE ";B
40 INPUT "BATTI IL MESE ATTUALE ";N
50 LET V = ((B-A) *12) + (N-M)
60 PRINT
70 PRINT "TU HAI VISSUTO ";V; " MESI"
80 GO TO 10
```

3. Programma per il calcolo dei mesi vissuti.

```
  10 PRINT "DIMOSTRAZIONE"
  20 GO SUB 1000
  30 PRINT "DI USO"
  40 GO SUB 1000
  50 PRINT "DELLE SUBROUTINE"
  60 GO SUB 1000
  70 STOP
1000 PRINT "***********************************"
1010 PRINT
1020 RETURN
```

4. Dimostrazione di uso delle subroutine.

COSA È UN LINGUAGGIO DI PROGRAMMAZIONE?

Per rispondere all'arduo quesito posto dal titolo occorre prima chiarire che cosa è in realtà un computer.

Quanti di voi possiedono già un computer (ZX S. o altri) si saranno sicuramente resi conto della assoluta incapacità della macchina a svolgere un qualsiasi compito senza l'ausilio di un programma, sia esso in BASIC, linguaggio macchina o altro. I computer in effetti sono dei magnifici "idioti", essi infatti possiedono solo una notevole memoria e possono eseguire dei calcoli elementari (somme, sottrazioni, spostamenti di dati, ecc.) a velocità incredibili, dell'ordine di decimillesimi di secondo o meno ancora.

LE FUNZIONI DELLA CPU

Tutte le operazioni svolte da un computer sono presiedute dalla CPU, la quale, aiutata da altri chip, esegue gli ordini che gli vengono impartiti. In linea generale l'efficienza di una CPU dipende dalla sua velocità e dal numero di istruzioni che è in grado di eseguire, ma di questo discuteremo più avanti, quando inizieremo a parlare di linguaggio macchina: per ora sorvoliamo il discorso CPU e concentriamoci invece sui diversi interpreti o compilatori disponibili per i linguaggi ad alto livello.

Iniziamo con il precisare cosa si intende per linguaggi ad alto livello e come questi ultimi si possano suddividere in compilati e interpreti; parleremo in seguito dei vantaggi e degli svantaggi degli uni rispetto agli altri.

Come molti di voi sapranno, un computer non è assolutamente in grado di capire direttamente ciò che voi volete dirgli, esso infatti comprende solo un complesso linguaggio formato da numeri, i quali, combinandosi fra loro, formano delle istruzioni elementari comprensibili per la CPU. Questo insieme di numeri costituisce il LINGUAGGIO MACCHINA (l/m) di un computer, o meglio, della CPU del computer.

La maggior parte dei personal computer incorpora un sistema operativo, il quale non è altro che un lungo programma scritto interamente in linguaggio macchina e generalmente posto in ROM; questo programma gestisce praticamente ogni attività svolta dal computer e in molti casi comprende anche un interprete, o compilatore, per un determinato linguaggio (BASIC, Pascal, ecc.).

Un interprete è un pezzo più o meno lungo di programma in codice macchina in grado di riconoscere le istruzioni di quel determinato linguaggio poste in un programma e eseguire, per ognuna di esse, una corrispondente serie di istruzioni in l/m.

Ovviamente, ogni volta che farete girare il vostro programma, l'interprete dovrà confrontare di nuovo ogni istruzione che trova con il vocabolario che possiede e, dopo averla riconosciuta, eseguire il suo "alter ego" in

l/m. È comprensibile come tutto ciò comporti un notevole rallentamento di esecuzione dei vostri programmi, specialmente se il linguaggio che usate dispone di un vocabolario piuttosto vasto di comandi. Per ovviare a questo problema sono stati ideati degli interpreti compilatori. Questi ultimi non ritraducono ogni volta le istruzioni del programma, ma lo fanno una sola volta. Facendolo girare la prima volta il programma viene infatti completamente trasformato nel suo equivalente in linguaggio macchina. Quando poi voi farete eseguire il programma, non occorreranno interpreti poiché, essendo ormai in l/m, è già direttamente comprensibile dalla CPU e non neccesita quindi di ulteriori manipolazioni.
Purtroppo un programma compilato non può più essere modificato e spesso in caso di errori occorrerà riscriverlo tutto. È comprensibile il motivo per cui la stragrande maggioranza dei personal computer adottino un linguaggio interprete quale il BASIC, permettendo una interattività con l'utente molto maggiore di quella possibile con un qualunque sistema a compilatore.

L'ORIGINE DEL BASIC

Chiarito il discorso in generale, passiamo a esaminare più in dettaglio quali sono le caratteristiche dei linguaggi più diffusi, per cosa si differenziano e a quali applicazioni sono più portati.
Cominciamo naturalmente con quello che si può giustamente considerare il portabandiera dei linguaggi per computer, il più conosciuto, il più semplice e il più discusso: parliamo ovviamente del BASIC.
Il BASIC nella sua prima versione risale al "lontano" 1964. Esso è stato espressamente concepito come un linguaggio molto semplice sia da apprendere che sa usare; per questo motivo è ben presto diventato il linguaggio di programmazione più diffuso per personal computer.
Il BASIC non tende in particolare a nessun tipo di applicazione specifica benché possa risolvere in modo soddisfacente problemi di tipo scientifico-matematico. A causa della sorprendente diffusione di questo linguaggio sono nati un numero incredibile di dialetti, i quali pur mantenendo invariati i principali comandi, a volte differiscono molto dalla versione standard della Microsoft, rendendo problematico, se non proprio impossibile, l'adattamento di programmi scritti per un diverso modello di computer.

Prendendo comunque come riferimento il BASIC standard possiamo fare alcune considerazioni. La prima osservazione da fare riguarda l'organizzazione stessa del BASIC che rende molto difficile seguire i canoni della programmazione strutturata; i molti salti contenuti in un programma ne rendono difficile l'interpretazione, l'uso delle procedure è piuttosto limitato e l'assegnazione delle variabili è spesso troppo permissiva.
Inoltre è difficile trattare grandi quantità di dati.
Bisogna comunque tenere presente che, come abbiamo già detto, il BASIC è un linguaggio

rivolto ai principianti e non ha quindi eccessive pretese.

UN ALTRO LINGUAGGIO: IL PASCAL

Abbandoniamo momentaneamente il BASIC per fare la conoscenza di un altro linguaggio di programmazione molto diffuso: il Pascal.
Nato anch'esso diversi anni fa, il Pascal è un linguaggio molto ben fatto, anche se di non facile apprendimento. Il suo creatore aveva chiaramente idea di sviluppare un linguaggio che costringesse alla programmazione strutturata; ciò che ne è risultato è veramente notevole dal punto di vista della chiarezza e leggibilità. Per quanto riguarda la stesura di programmi complessi è necessario procedere a blocchi e sovente bisogna perdere gran parte del tempo a correggere parti di programma o procedure nelle quali si erano dimenticati dei parametri o si erano usate variabili non ammesse. Il Pascal è infatti molto rigido riguardo l'uso delle variabili che verranno usate e devono tutte essere dichiarate preventivamente. Nel complesso il Pascal, nonostante alcune limitazioni, è comunque un ottimo linguaggio, anche se il suo impatto su quanti sono abituati al BASIC potrebbe essere un po' "traumatico".

IL FORTH E IL COBOL

Dopo avere parlato di BASIC e Pascal non potevamo certo tralasciare il Forth, il linguaggio della quarta generazione.
Il miglior modo per descrivere il Forth consiste nel dire che è un linguaggio "anticonformista", esso si differenzia infatti in modo notevole dalla maggioranza degli altri linguaggi.
Per ovvi motivi di spazio non ci soffermeremo a spiegarvi nei dettagli quale è la sua struttura e come si sviluppano i programmi. Vi basti sapere che disponendo di un compilatore Forth con un vocabolario di base è possibile risolvere praticamente ogni problema di programmazione. Il Forth è in sostanza un linguaggio molto potente benché all'inizio il suo uso non risulti molto semplice. Chi volesse approfondire l'argomento può consultare uno dei tanti libri in commercio.
Un tipico linguaggio nato per una specifica applicazione è il COBOL.
Come si può facilmente dedurre dal nome (COmputer Businnes Oriented Language), il COBOL è un linguaggio creato specificatamente per usi gestionali. Esso è stato sviluppato basandosi sull'esperienza maturata dai precedenti tipi linguaggi meno evoluti.
La costituzione stessa di questo linguaggio ne rende proibitivo l'uso sui piccoli sistemi, mentre è il preferito per applicazioni commerciali su computer di stazza superiore.

LISP: UN LINGUAGGIO RIVOLUZIONARIO

Sorvolato velocemente anche il COBOL, passiamo a un linguaggio veramente unico e senza dubbio rivoluzionario: il LISP.
Creato intorno alla fine degli anni '50, il LISP rappresenta uno dei pochi esempi funzionali di linguaggi orientati verso applicazioni di intelligenza artificiale. La struttura di questo lin-

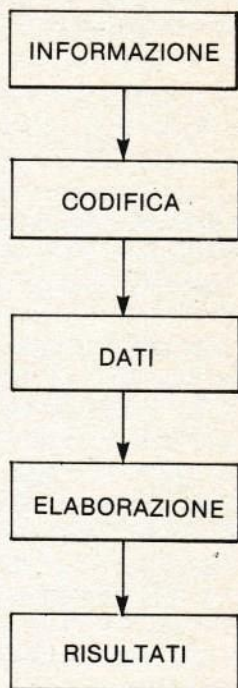
guaggio è piuttosto complessa e per poterne sfruttare appieno le notevoli caratteristiche occorre una buona esperienza di programmazione e una perfetta conoscenza dello stesso. Il LISP ha poche funzioni di tipo matematico, la sua forza è data infatti dalla straordinaria flessibilità nel manipolare i dati. Esistono diverse versioni di questo linguaggio per i più conosciuti personal ma, date le sue caratteristiche, non è molto diffuso e l'uso è limitato a poche applicazioni, prevalentemente a scopo di studio.

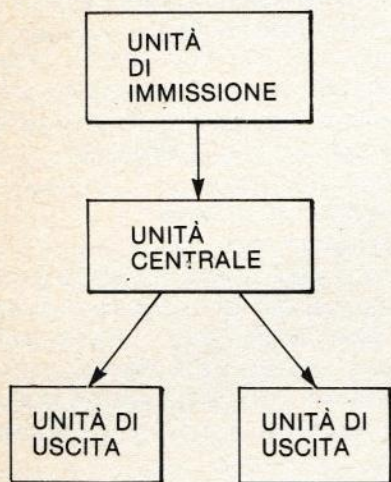
FORTRAN IL CAPOSTIPITE

Non potevamo concludere la nostra carrellata sui più noti linguaggi di programmazione senza parlare del loro capostipite, il FORTRAN. La prima delle tante versioni del FORTRAN è del 1954; esso venne concepito per risolvere problemi di tipo matematico ma presentava numerose carenze e, nonostante i vari aggiornamenti a cui venne sottoposto, si dimostrò pur sempre poco efficiente e venne in buona parte sostituito, o affiancato, da linguaggi più progrediti. Terminiamo qui la rassegna in verità piuttosto limitata, siamo sicuri che gli estimatori di linguaggi meno diffusi quali il Logo, il Pilot, l'APL, ecc., ci perdoneranno per non averli citati in questo articolo che non voleva certo fornire un giudizio definitivo su ogni linguaggio preso in esame, ma intendeva solo chiarire cosa si intende per linguaggio di programmazione e esaminare a grandi linee alcuni dei più diffusi di essi.

Speriamo di avervi chiarito un po' le idee sull'argomento e consigliamo, a quanti di voi sono interessati, di approfondire la conoscenza dei numerosi linguaggi disponibili, ciò permetterà anche di accrescere la vostra abilità nell'arte della programmazione.

PARLIAMO DI INFORMATICA...

La parola informatica è un neologismo di origine francese con il quale si intende riferirsi all'elaborazione dell'informazione con sistemi automatizzati.
Informatica deriva dalla parola informazione, significato attribuito dall'uomo ai dati secondo convenzioni prestabilite. L'elemento base di qualsiasi sistema di comunicazione è costituito dall'informazione. L'ufficio è la parte di un'impresa principalmente finalizzata all'acquisizione, memorizzazione, reperimento, distribuzione dell'informazione, mediante opportune strutture e processi, allo scopo di prendere nel migliore dei modi le decisioni relative alla vita aziendale.
Il mezzo tecnico che consente l'automazione dell'organizzazione e del trattamento dei dati è l'elaboratore elettronico. È perciò importante conoscerne i principi di funzionamento per poter essere in grado di fare delle scelte razionali e per cercare di utilizzarlo nel migliore dei modi.
Si era venuto a credere, e taluni lo credono tuttora, che il calcolatore elettronico sia una sorta di "cervello diabolico" difficilissimo da comprendere ed usare; in realtà il calcolatore, pur essendo un insieme piuttosto complesso di circuiti, è in grado di svolgere solo funzioni elementari.
L'unica differenza è che la macchina è in grado di elaborare i dati ad altissima velocità e con ridottissime possibilità di errore. La macchina non è in grado di ragionare con una propria logica, siamo noi ad istruirla e guidarla passo a passo mediante la programmazione.
Affinché sia possibile l'elaborazione delle informazioni, è necessario che queste vengano opportunamente codificate in modo tale che possano venire facilmente comprese dall'elaboratore.

L'elaborazione dei dati prende il nome di:
ELABORAZIONE MANUALE: quando si realizza con l'ausilio di normali macchine da scrivere e di calcolatrici da tavolo più o meno sofisticate.
ELABORAZIONE SEMIAUTOMATICA: quando l'impiego del calcolatore avviene con fre-

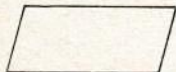
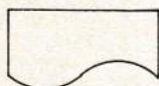
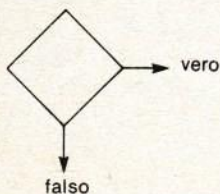
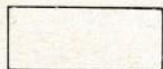
quenti intervalli umani a livello decisionale.
ELABORAZIONE AUTOMATICA: quando si realizza con l'impiego di elaboratori elettronici e l'intervento umano avviene semplicemente a livello manuale.
Un sistema elettronico per l'elaborazione dei dati è composto da una UNITÀ CENTRALE e da alcune UNITÀ PERIFERICHE di I/O per l'immissione e l'emissione dei dati.
(Es. unità di immissione: lettore di schede, unità a dischi, unità a nastro. Es. unità di emissione: perforatore di schede, registratori a nastro magnetico o disco, stampante, unità video, plotter).

L'unità di comando (centrale) coordina l'intervento della memoria centrale delle memorie ausiliarie e dell'unità aritmetica e logica.
Il pallottoliere o abbaco fu per molto tempo l'unico sistema per risolvere calcoli. L'invenzione dei logaritmi da parte di Nepero nel 1614, e più tardi quella del regolo calcolatore, portarono un notevole impulso ed aiuto allo svolgimento mentale dei calcoli.
Nel 1642 Blaise Pascal inventò una vera e propria macchina calcolatrice capace di fare addizioni. Questa macchina funzionava con un metodo molto simile a quello comunemente adottato dalle macchine calcolatrici moderne meccaniche. In seguito, nel 1671, Gottfried Leibniz disegnò e realizzò una macchina calcolatrice capace di addizionare e moltiplicare meccanicamente. Molti anni più tardi, nel 1890, l'americano Hollerith applicando l'elettricità alle macchine calcolatrici segnò il primo passo verso il moderno computer elettronico.
Bisognerà aspettare ancora cinquanta anni per arrivare alla realizzazione, negli Stati Uniti, del primo computer elettromagnetico. Due anni più tardi infine, nel 1946, sempre in America venne realizzato il primo computer elettronico, il quale è stato col passare del tempo sempre più migliorato fino ad arrivare ai moderni sistemi di elaborazione dei dati.

GLI ALGORITMI

Affinché sia possibile risolvere un problema si deve sviluppare un metodo di soluzione o algoritmo capace di realizzare in modo esplicito e non ambiguo, passo dopo passo, la trasformazione dei dati fino ad arrivare alla soluzione del problema assegnato.
Un algoritmo è quindi una successione finita di istruzioni che possono essere eseguite da un calcolatore producendo risulta-

ti da dati assegnati.
Un algoritmo può essere rappresentato graficamente mediante un diagramma a blocchi o flowchart.
I simboli utilizzati per eseguire un diagramma a blocchi sono i seguenti:

Istruzioni di inizio e fine esecuzione

Operazione di input

Operazioni di output

Operazione di condizione o verifica

Istruzione di assegnazione

Istruzione di salto incondizionato

Le frecce → indicano la direzione dell'istruzione successiva.
Gli algoritmi devono soddisfare i seguenti requisiti:
a) Generalità - un algoritmo deve essere applicabile a qualsiasi insieme di dati appartenenti ad una prefissata categoria o dominio. (Si chiama dominio l'insieme di tutti i dati a cui si applica un determinato algoritmo)
b) Finitezza - un algoritmo deve essere costituito da una successione di istruzioni che possono essere eseguite dal calcolatore un numero finito di volte.
c) Non ambiguità - un algoritmo non deve essere costituito da istruzioni che si contraddicono a vicenda oppure che portino a qualche paradosso.
I valori su cui operano le istruzioni di un linguaggio di programmazione per produrre nuovi valori possono essere:
a) numerici
b) logici: vero-falso
c) alfanumerici
Quando un valore è noto a priori, ed è sempre lo stesso in tutte le sue applicazioni, esso può venire assegnato tramite una costante che potrà perciò essere numerica, logica o alfanumerica.
Le costanti alfanumeriche vengono pure dette stringhe, i valori alfanumerici vengono sempre riportati fra virgolette.
Le variabili servono come portatrici di valori: un valore può essere assegnato ad una variabile e questo rimane associato ad essa finché non intervenga una nuova assegnazione di valore alla medesima variabile. Tra due assegnazioni successive il

valore di una variabile rimane immutato. Una variabile può essere di tipo numerico, logico, alfanumerico.
Le istruzioni fondamentali di ogni linguaggio di programmazione, individuabili con un nome o numero, sono dei seguenti tipi:
a) istruzioni di assegnazione che permettono di assegnare il valore di una qualunque variabile;
b) istruzioni di salto incondizionato che servono ad interrompere il normale ordine di esecuzione delle istruzioni in un algoritmo;
c) istruzione di condizione che, in base al confronto di due valori, condizionano l'esecuzione di certe istruzioni rispetto ad altre;
d) istruzioni di trasmissione (o di input-output) che permettono di trasferire valori fra il mondo esterno e il calcolatore o viceversa;
e) istruzioni di inizio di esecuzione e di fine esecuzione che comandano, rispettivamente, di iniziare e di arrestare l'esecuzione delle istruzioni che costituiscono l'algoritmo.
Eccetto che per le istruzioni di salto incondizionato, le istruzioni di un algoritmo vengono eseguite nell'ordine in cui sono scritte e l'esecuzione di una istruzione non inizia sino a che non sia terminata l'esecuzione dell'istruzione precedente.
Una parte fondamentale degli algoritmi è il ciclo.
Per ciclo si intende una particolare successione di istruzioni che permettono ad un gruppo di istruzioni di essere eseguite ripetutamente dal calcoaltore sino a che non si siano verificate alcune condizioni che alterano l'ordine di esecuzione della successione di istruzioni costituenti il ciclo.
Un ciclo è costituito da istruzioni che permettono di inizializzare il ciclo, dal gruppo di istruzioni che devono essere eseguite ripetutamente (istruzioni fondamentali del ciclo), da istruzioni che alterano l'ordine di esecuzione della successione di istruzioni (istruzioni di controllo) e infine da istruzioni che modificano il valore di alcune grandezze su cui operano le istruzioni fondamentali del ciclo (istruzioni di modifica).

Claudio Pozzoni

DATI: LO SPECTRUM LI MEMORIZZA COSÌ

Probabilmente molti di voi non si sono mai curati di capire come lo Spectrum (o qualunque altro computer a 8 bit) immagazzina e elabora le informazioni che voi inserite tramite la tastiera.
In realtà non vi è nulla di complesso all'interno del vostro Spectrum, anzi, è tutto di una semplicità sorprendente.
Come abbiamo già detto, lo spectrum è un computer a 8 bit, ciò significa che qualunque dato deve essere memorizzato sotto forma di byte, cioè numeri binari composti appunto da 8 singole cifre binarie; ognuna di queste cifre corrisponde a un bit.
Anche se all'apparenza il sistema binario può sembrare completamente diverso dal decimale che siamo abituati a usare, basta approfondire un poco il discorso per rendersi conto che in realtà tutti i sistemi di numerazione si somigliano molto.
Usando il sistema decimale, per calcolare il massimo valore rappresentabile con un determinato numero di cifre è sufficiente elevare 10 (la base del sistema) al numero di cifre impiegate e sottrarre uno dal risultato.
Per esempio, per trovare il più alto intero visualizzabile con 3 cifre decimali procediamo nel modo seguente:

10 ↑ 3 = 1000 - 1 = 999

Semplice vero?!?
Lo stesso discorso vale anche per il sistema binario, cambiando naturalmente la base che non sarà più 10 ma 2. Quindi sapendo che un byte è composto da 8 cifre (bit) possiamo ricavare il massimo intero che esso può contenere.

2 ↑ 8 = 256 - 1 = 255

Ogni bit può dunque contenere qualunque numero compreso tra 0 e 255.
Ma allora come fa il computer a rappresentare una locazione di memoria che nel caso di un sistema a 64 K (come lo spectrum, infatti 16 ROM + 48 RAM = 64), può arrivare fino a 65535?
Semplice! Può farlo unendo 2 bytes, nel qual modo avrà a disposizione non più 8 ma bensì 16 bit per rappresentare un qualunque valore numerico.
Rifacciamo quindi il calcolo eseguito precedentemente.

2 ↑ 16 = 65536 - 1 = 65535

Capito il trucco?
Ora però sorge un piccolo problema in quanto la memoria dello spectrum è comunque organizzata sotto forma di bytes, quindi per memorizzare un numero a 16 bit dovrà essere scomposto in due da 8 bit, quindi verrà memorizzata prima la parte meno significativa e poi quella più significativa.
CHIARO??? Spero di sì, perché nel prossimo numero vi aspettano cose davvero molto interessanti, a patto che prima assimiliate bene i concetti esposti in questo articolo.

TI SUPER SOUND

È indubbio che nell'ultimo decennio l'elettronica, e di conseguenza l'informatica, abbiano avuto uno sviluppo senza precedenti.
Ormai può sembrare assurdo non trovare un microprocessore nel ferro da stiro o un megacomputer con tanto di sintetizzatore vocale, incorporato nella cucina nuova. Certo è facile abituarsi a queste cose, come è stato facile abituarsi all'energia elettrica, al telefono o alla televisione, senza per questo chiedersi come questi funzionino o chi li abbia ideati e costruiti.
Certamente qualunque nuova invenzione o scoperta al suo debutto incontra sempre una certa dose di incredulità e scetticismo. Lo stesso è stato per la rivoluzione portata dai circuiti integrati e, successivamente, dai microprocessori. Per questo motivo nei primi tempi può essere molto difficile trovare delle aziende disposte a rischiare con un nuovo prodotto che potrebbe fare facilmente la loro fortuna o, altrettanto facilmente mandarle in rovina. Dipende quindi da una scelta iniziale la futura affermazione delle case produttrici in un determinato campo e il prestigio di cui esse godranno.
Senza dubbio una delle più prestigiose produttrici di materiale elettronico ad alto grado di integrazione è l'americana TEXAS INSTRUMENTS presente fin dall'inizio nel campo dell'elettronica e dell'informatica; pochi di voi lo ricorderanno, ma essa produceva calcolatrici portatili quando la concorrenza si impegnava ancora a ideare nuovi tipi di pallottolieri o al più qualche avveniristico modello di regolo calcolatore.
Immaginatevi cosa succede quando un colosso del genere decide di inserirsi nel fiorente mercato degli home computer!!!
Una macchina prodotta da una tale azienda sarà sicuramente all'avanguardia, tanto da surclassare letteralmente ogni eventuale avversario.
Questa macchina esiste. Si tratta, come avrete capito, del TI 99/4-A.
Considerazioni estetiche a parte, quando si lavora con il TI 99 si ha realmente l'impressione di dominare un computer molto potente, in grado di esaudire qualunque desiderio. Una tale macchina merita sicuramente un'analisi molto approfondita che ci impegnamo a continuare successivamente.
In questo articolo esamineremo una caratteristica sicuramente molto accattivante di questo straordinario computer: il suono.
Molti computer di recente uscita hanno capacità sonore simili a quelle del TI 99, il quale si può però meritatamente considerare il capostipite per eccellenza dei nuovi home computer, o almeno del modello ideale di home computer, con buone capacità sia grafiche che sonore e di facile programmazione.
Passiamo ora a esaminare più in dettaglio le possibilità sonore del TI.

Esso possiede un'unica istruzione per la generazione di effetti sonori, ma è talmente flessibile da permettere di creare una vastissima gamma di suoni a diversi livelli di volume.
La sintassi è la seguente:
CALL SOUND (durata, frequenza 1, volume 1, frequenza 2, volume 2...)
Per ogni istruzione SOUND possono essere emessi fino a 4 suoni simultaneamente, definendo per ognuno la frequenza e il volume, mentre la durata è uguale per tutti e viene definita come primo parametro dell'istruzione. Essa può assumere un valore da 1 a 250, in cui il valore minimo corrisponde a 1 millisecondo mentre il massimo è di 4,25 secondi.
Ogni frequenza corrisponde a un determinato suono che sarà tanto più acuto quanto più alta è la frequenza selezionata. Naturalmente è possibile trovare una corrispondenza fra le note musicali e la rispettiva frequenza. Qui di seguito riportiamo una tabella delle frequenze relativa a ogni nota della scala per due ottave, a voi il compito di completarla.

LA	110	220
LA#,SI	117	233
SI	123	247
DO	131	262
DO#,RE	139	277
RE	147	294
RE#,MI	156	311
MI	165	330
FA	175	349
FA#,SOL	185	370
SOL	196	392
SOL#,SI	208	415

L'ultimo parametro richiesto per la produzione di un suono è il volume. Quest'ultimo può variare da 0 (forte) a 30 (debole); naturalmente esso dipenderà anche da come avete regolato il volume sul televisore (consigliamo comunque un valore medio).
Tutto qui? Naturalmente no! Il nostro fido TI ha sempre qualche sorpresa in serbo per i suoi affezionati amici.
Tanto per cominciare, se mettiamo al posto della frequenza un valore negativo compreso tra -1 e -8, otteniamo l'emissione di un "rumore"; questa caratteristica è molto utile e si presta in modo particolare per applicazioni di tipo "giocoso" che sono probabilmente quelle che molti di voi prediligono.
A questo punto vorremmo parlarvi delle molteplici applicazioni offerte dalla possibilità di emissione di più suoni in contemporanea, ma preferiamo che siate voi stessi a scoprirle... e se riuscite a fare qualcosa di interessante non mancate di farcelo sapere.
Per chi desidera invece qualcosa di pronto ecco un bel programma che vi permetterà (dopo un po' di pratica) di suonare qualsiasi melodia, a patto che sappiate interpretare perlomeno lo spartito musicale. In caso contrario potrete comunque usarlo come divertente programma per generare sequenze di suoni a vostro piacimento.

```
10 CALL CLEAR
15 DIM FA (13)
20 DIM NA (11)
30 INPUT "VOLUME 0-30?":
   VOL
40 INPUT "DURATA?":DUR
50 FOR A=1 TO 13
```

```
60  READ FA (A)
70  NEXT A
75  DATA 220, 233, 247, 262,
    277, 294, 311, 330, 349,
    370, 392, 415, 440
90  RESTORE 145
100 FOR A = I TO 11
110 READ NA (A)
120 K = NA (A)
130 GOSUB 200
140 NEXT A
145 DATA 4, 4, 4, 6, 8, 6, 4, 8,
    6, 6, 4
150 END
200 CALL SOUND (DUR,FA
    (K),VOL)
205 RETURN
```

La linea 15 dimensiona una matrice di 13 variabili atte a contenere tutte le note della seconda ottava (vedi tabella). Naturalmente potete aumentare il numero di variabili per la matrice FA e aggiungere altri valori nell'istruzione DATA di linea 75.
La linea 20 dimensiona una seconda variabile, la quale dovrà contenere il numero della nota che si desidera suonare. In questo caso è stata predisposta per contenere il valore di 11 note, ma alterandola opportunamente e cambiando di conseguenza i valori dei DATA di linea 145 potrete fargli contenere qualunque brano, anche molto lungo.
Le linee 30 e 40 servono a selezionare il volume desiderato e la durata di ogni nota. Le linee successive fino alla 75 assegnano i valori delle varie note della scala alla matrice destinata a contenerli.
Le linee 90 e 100 iniziano un ciclo della lunghezza del numero di note da suonare. La linea 110pone in NA il valore della nota letto nella data di linea 145 e, successivamente, assegna tale valore alla variabile K, dopodiché rinvia alla subroutine di linea 200, la quale emette una nota del volume e della durata scelti, con una frequenza corrispondente alla nota numero K contenuta nella matrice FA. La linea 150 evidenzia la fine del programma.
Semplice vero?!?
Comunque se non siete sicuri di aver capito bene quanto esposto in questo articolo, fate delle prove, magari modificate il programma che vi abbiamo proposto o createne di migliori voi stessi, l'importante è che assimiliate bene i concetti fondamentali che stanno alla base di una buona programmazione su questo ineguagliabile computer che, un po' alla volta, si rivelerà uno strumento insostituibile, sempre al vostro servizio.

M. C.

ISTRUZIONI PER LA CASSETTA COMPUTING VIDEOTECA N. 3

COME INIZIARE

(Lato B: TI 99/4A. Le spiegazioni dei programmi appariranno direttamente in video).

Per caricare i programmi inserite la cassetta nel registratore e scrivete OLD CS1 seguito dal tasto ENTER. Apparirà la scritta REWIND CASSETTE TAPE, THEN PRESS ENTER, premete quindi il tasto ENTER, apparirà ora il messaggio PRESS CASSETTE PLAY THEN PRESS ENTER che significa: premi il tasto PLAY del registratore e quindi ENTER: avviate dunque il registratore in play e premete di nuovo ENTER.

Quando il computer avrà terminato di caricare il primo programma fermate il registratore e scrivete RUN seguito come sempre da ENTER per lanciarne l'esecuzione.

Quando avrete terminato di usare un programma resettate il computer spegnendolo per un momento, dopodiché ripetete le operazioni di caricamento sopradescritte.

(Lato A: ZX Spectrum)

Come ben sapete, COMPUTING VIDEOTECA vuole offrire il meglio ai suoi lettori, presentando programmi e articoli utili e divertenti.

Purtroppo i tempi di caricamento dei programmi da nastro sono piuttosto lunghi e spesso si è costretti a passare diversi minuti davanti allo schermo muto. Da ora in poi ciò non avverrà più, almeno per i programmi di COMPUTING, abbiamo infatti messo a punto un efficiente sistema di memorizzazione dei programmi su nastro che riduce drasticamente i tempi di caricamento, sicuri di fare cosa gradita alle migliaia di appassionati che fedelmente ci seguono.

ISTRUZIONI DI CARICAMENTO

Per caricare i programmi inserite la cassetta nel registratore e scrivete LOAD "" (le " si ottengono con SYMBOL SHIFT e P), su-

bito dopo premete il tasto ENTER e avviate il registratore in PLAY.

A questo punto basterà seguire le istruzioni che appariranno sul video.

Quando avrete finito di usare un programma spegnete il computer staccando per un momento il cavetto di alimentazione. Battete di nuovo LOAD "" seguito da ENTER per caricare il programma successivo.

VISISCHOOL
LA SCUOLA IN UN DIAGRAMMA

Capire con un solo colpo d'occhio tutto l'andamento del proprio anno scolastico può essere utilissimo. Può, ad esempio, mettere al riparo da brutte sorprese o quanto meno prevenire cedimenti o aiutare a incrementare lo studio nelle materie giuste al momento giusto.

Con Visischool, un programma che permette di raggiungere risultati possibili, fino a qualche tempo fa, solo alle grandi aziende capaci di analizzare in ogni momento i propri diagrammi di produzione, ogni studente ha la possibilità di esaminare sul video gli istogrammi del proprio andamento scolastico.

Il programma, realizzato sullo Spectrum, è un vero gioiello, nel suo genere, in grado di non sfigurare (anzi!...) di fronte ai più sofisticati programmi per computer.

È semplicissimo da usare, memorizza tutti i voti nelle diverse materie di studio e permette di visualizzare istantaneamente un grafico a istogrammi.

Dopo aver caricato il programma, sul video appare il menù principale costituito da cinque opzioni selezionabili premendo il numero corrispondente. Se usate Visischool per la prima volta dovete scegliere la opzione 3 che permette l'inserimento delle materie da considerare.

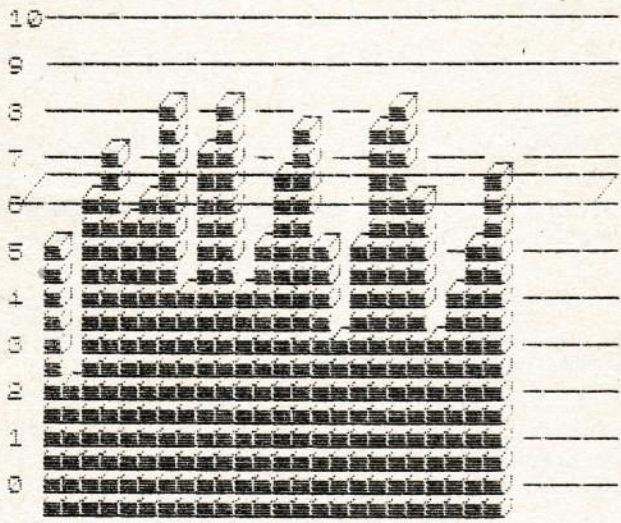
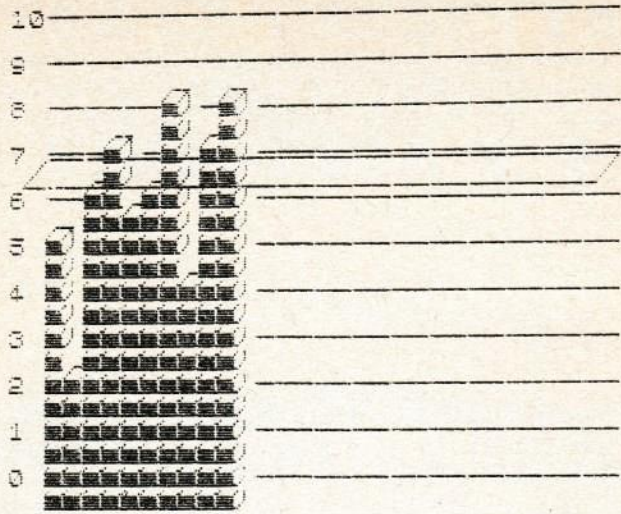
Ogni volta che vorrete aggiungere nuovi voti, selezionate la opzione 1 e digitate il nome della materia da aggiornare.

La seconda opzione mostra, sotto forma di grafico, l'andamento dei voti, e quindi il rendimento scolastico, nella materia scelta.

La quarta e la quinta opzione permettono di caricare e salvare la situazione scolastica su nastro.

Visischool è un programma particolarmente indicato per coloro che vogliano scoprire utilizzazioni pratiche del computer nella vita di ogni giorno, cominciando dalla scuola per proseguire negli infiniti campi che la fantasia ed il tempo portano ad esplorare.

10
9
8
7
6
5
4
3
2
1
0

10
9
8
7
6
5
4
3
2
1
0

VISI-SCHOOL++©1984 E.Video AVS

```
INSERIMENTO VOTI....1
GRAFICO.............2
INS. MATERIE........3
SALVATAGGIO DATI....4
CARICAMENTO DATI....5
```

BUIO!

Vi proponiamo stavolta qualcosa di decisamente insolito, un gioco, ma non un comune gioco d'azione, bensì un sofisticato ADVENTURE, una lunga e difficile avventura nella quale vi trovate soli con voi stessi, in un immenso castello di ben 16 piani. Lo scopo del gioco consiste nel trovare un tesoro nascosto nell'ultimo piano del castello in cui siete rinchiusi. Ma non pensate che trovare tesori sia così facile, dovrete infatti lottare contro orribili mostri che cercheranno in tutti i modi di trasformarvi in una sostanziosa colazione (o cena, a seconda dell'ora in cui li incontrate).

In ogni livello dovrete cercare di recuperare la chiave che vi permetterà di passare al successivo, a condizione che disponiate di un certo numero di monete d'oro (2500 al primo livello). La chiave è rappresentata sulla mappa dal simbolo +. A proposito, avete la possibilità di consultare la mappa premendo il tasto M, ma solo dopo aver ucciso almeno 5 mostri.

In ogni livello si trova anche un'arma, rappresentata sulla mappa dalla lettera A: sarà bene che la troviate poiché essa aumenterà la vostra forza e vi sarà molto utile per affrontare i numerosi mostriciattoli che bazzicano per il castello.

Se verrete attaccati da un mostro potrete attaccarlo a vostra volta premendo il tasto A, oppure ritirarvi premendo R; se decidete di attaccarlo potete scegliere se colpirlo alle gambe, al corpo o in testa, premendo rispettivamente i tasti G, C e T. I tasti si dovranno mantenere premuti per tutta la durata del combattimento. Ogni volta che il mostro vi colpisce, il bordo del video diverrà bianco.

Osservando la mappa noterete anche degli strani simboli (#), essi rappresentano le stanze del mistero, entrandovi potrete trarne beneficio o restarne vittima; in ogni caso dopo che le avrete usate esse spariranno.

Per concludere vi ricordiamo che per muovervi dovrete usare i tasti di cursore (5/6/7/8) e la vostra posizione sarà segnalata sulla mappa da un quadratino lampeggiante.

Per salvare su nastro la situazione attuale premete il tasto CAPS SHIFT insieme a S. Per ricaricarla successivamente basterà usare LOAD "".

Siamo certi che anche se si tratta di un genere di gioco inconsueto, Buio incontrerà in breve tempo i vostri favori, poiché permette un coinvolgimento decisamente superiore a qualunque ARCADE e vi avvincerà dall'inizio alla fine (non vostra spero.). Buon divertimento!

ANATOMIA

Sicuramente la maggior parte di noi conosce molto poco il proprio corpo, spesso infatti capita di non sapere il nome di alcune parti e in questi casi ben pochi si preoccupano di consultare un libro per cercare di approfondire la loro conoscenza.

Ma ora, grazie a questo straordinario programma, potrete rapidamente conoscere la dislocazione dei vari organi e le loro caratteristiche essenziali.

Una volta caricato il programma vi verrà presentato un menù con 5 opzioni. Se usate il programma per la prima volta oppure volete aggiungere altri organi, scegliete la prima opzione che vi permetterà appunto di inserire i dati che vi interessano.

Una volta selezionata l'opzione 1 vedrete comparire sullo schermo un corpo umano, suddiviso nelle sue parti principali; al centro dello schermo si trova un quadratino nero. Muovete il quadratino con i tasti di cursore (5/6/7/8) e, quando sarete sul punto desiderato premete il tasto 0. Vi verrà chiesto il nome dell'organo, osso o arto indicato dal quadratino e, se lo desiderate, alcune note su di esso. Il nome e le note non dovranno comunque superare i 20 caratteri di lunghezza.

La seconda opzione permette di identificare un organo semplicemente scrivendo il suo nome: se esso è stato memorizzato apparirà la sua collocazione nel corpo e le sue caratteristiche.

L'opzione numero 3 è molto utile, in quanto può servire per modificare dei dati che non sono stati introdotti correttamente.

Per concludere, la quarta e quinta opzione consentono rispettivamente di salvare e caricare da nastro i dati introdotti. Sarebbe bene che voi, dopo avere inserito i dati (magari con l'ausilio di un buon libro), li salvaste su una cassetta a parte, così prima caricherete il programma principale e dopo, con l'opzione 4, caricherete i dati a esso relativi per una semplice e rapida consultazione.

BRAIN

Brain, il cervello, il genio o comunque vogliate chiamarlo, è un programma che metterà a dura prova la vostra intelligenza e i vostri nervi.

Certo non è facile descrivere un programma come questo e il miglior suggerimento che si può dare è di provarlo. Cercheremo comunque di spiegare sommariamente come funziona e cosa fa.

Appena caricato il programma vi verranno fatte alcune domande a cui dovrete rispondere con SI o NO, oppure descrivendo l'oggetto che voi volete inserire. Successivamente vi saranno rifatte le stesse domande più altre riguardanti gli oggetti appena inseriti. Procedendo per deduzione, il computer scoprirà, in base alle risposte da voi fornite, una parola a cui avrete pensato, di cosa si

tratta, oppure, se quello che avete descritto non esiste, vi sarà chiesto di descriverne una sua caratteristica e il nome.

Alla fine di ogni "tornata" di domande il programma riprenderà dall'inizio, mostrando anche la memoria finora occupata dalle definizioni.

Il nome dell'oggetto e la sua descrizione non devono superare i 20 caratteri ciascuno, le eventuali eccedenze saranno impietosamente troncate.

Fate attenzione, appena caricato il programma inserite subito il CAPS LOCK (CAPS SHIFT & 2), in caso contrario comprometterestе il corretto funzionamento del programma.

Per salvare su nastro i dati da voi introdotti rispondete P alle domande, per caricare invece dei file precedentemente salvati rispondete con una L.

Buon divertimento e speriamo che il freddo non comprometta i vostri processi elucubrativi...

ESCAPE

Il gioco che vi proponiamo questa volta è veramente molto originale e appassionante.

Lo scopo consiste nel raggiungere il simbolo lampeggiante posto dalla parte opposta dello schermo, ma naturalmente molti ostacoli si frapporranno fra voi e lui, non ultimo un mostriciattolo malintenzionato che cercherà in tutti i modi di raggiungervi.

All'inizio di ogni partita disponete di tre vite che avete comunque la possibilità di aumentare durante il gioco. Per raggiungere la parte opposta del labirinto in cui vi trovate dovrete passare attraverso diverse porte che potrete aprire per mezzo delle chiavi di cui disponete e che sono indicate in basso a destra dello schermo. Oltre alle porte vi sono anche dei "pannelli" che sono di due tipi riconoscibili per il rumore che producono quando raggiunti; il primo tipo emette un suono acuto, vi regala una chiave e pochi punti, il secondo, che emette un suono più basso, ha l'effetto di aprire alcune porte e di chiuderne altre in modo pseudocasuale, oltre a darvi alcuni punti. Vi sono inoltre alcuni sacchi di denaro che, se raccolti, vi frutteranno diversi punti.

Una volta raggiunta la meta vi verranno dati in bonus i punti equivalenti al tempo risparmiato sul totale a disposizione che è rappresentato alla sinistra dello schermo da una colonna che cala progressivamente. All'inizio i secondi risparmiati vi frutteranno poco, ma in seguito, con l'aumentare del livello di difficoltà, aumenterà anche il loro valore.

Il cambio di quadro avviene in funzione del punteggio raggiunto, ma non vi diciamo altro per non privarvi della sorpresa.

Anche se in principio può sembrare un gioco fin troppo semplice non fatevi trarre in inganno, se raggiungerete punteggi elevati vi accorgerete di non avere la vita tanto facile...

Numeri già apparsi:

VIDEOTECA COMPUTER N. 1
per i possessori di Commodore 64

Nel manuale n. 1: I tasti funzionali del Commodore 64 - Pseudocodice e programmazione Basic - Il joystick.
Nella cassetta n. 1: Slalom - Slot Machine - Bilancio familiare - Briscola - Domino.

VIDEOTECA COMPUTER N. 2
per i possessori di Commodore 64

Nel manuale n. 2: Il basic più veloce - Una migliore gestione del video per il 64 - Disegnare con tastiera e joystick - Pseudocodice: 2ª lezione.
Nella cassetta n. 2: Tennis 3d - Totocalcio - Gestione magazzino - Wargame - Colour search.

VIDEOTECA COMPUTER N. 3
per i possessori di Commodore 64

Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Squash e Trampolino) - Conosci il tuo CBM 64?
Nella cassetta n. 3: Starway - Easyword - Poker - Forza 4 - Test Quoziente Intellettuale.

VIDEOTECA COMPUTER N. 4
per i possessori di Commodore 64

Nel manuale n. 4: Pseudocodice: Istruzioni read e data - Figura richiama - Grafico a barre. Il basic del CBM 64. I linguaggi di programmazione. Come personalizzare i programmi.
Nella cassetta n. 4: Dieta - Calorie dei cibi - Visischool - Sink - Hat in the ring.

PLAY ON TAPE N. 1
per i possessori di VIC 20

Nel manuale n. 1: I tasti funzionali del VIC 20 - Pseudocodice e programmazione Basic - Il joystick.
Nella cassetta n. 1: Totocalcio - Air attack - Cervellone - Inferno 3D - Bilancio familiare.

PLAY ON TAPE N. 2

Nel manuale n. 2: il basic più veloce - I caratteri speciali del VIC 20 - Disegnare con la tastiera e il joystick - Pseudocodice: 2ª lezione.
Nella cassetta n. 2: Test per misurare il Quoziente intellettuale - Easyword - Caccia al tesoro - Gestione Magazzino - Formula 1.

PLAY ON TAPE N. 3
per i possessori di VIC 20

Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Corsa automobilistica - Piranha) - L'interfaccia sconosciuta - Conosci il tuo VIC 20?
Nella cassetta n. 3: Sette e mezzo - Dieta - Calorie dei cibi - Gangster - Costi chilometrici.

COMPUTING VIDEOTECA N. 1
Per i possessori di Sinclair SPECTRUM ZX

Nel manuale n. 1: Pseudocodice e programmazione basic: 1ª e 2ª lezione - La gestione dei canali dello Spectrum - Come farsi una cassetta di "Subroutines".
Nella cassetta: Wargame - Gestione del magazzino - U.F.O. - Helibomber.

COMPUTING VIDEOTECA N. 2
per i possessori di Sinclair ZX SPECTRUM

Nel manuale n. 2: I cicli annidati del Basic - Come sviluppare un programma (Gara di sci) - Variabili interne e gestione del video nello Spectrum.
Nella cassetta n. 2: Frog Race - Pac Chian - Torre Laser - Dieta - Calorie dei cibi.

I numeri arretrati costano L. 15.000. Indirizzare vaglia o assegno a Editoriale VIDEO via Castelvetro 9 - 20154 Milano specificando il numero richiesto.
Ufficio tecnico e arretrati: telefono 02/3184829

Editoriale VIDEO:
COMPUTING VIDEOTECA n. 3

Direttore: Antonio Lucarella - Coordinamento tecnico: Roberto Treppiedi

Hanno collaborato: Marco Affer - Alberto Barbati - Massimo Cellini - Umberto Colapicchioni - Umberto Fassi - Alfredo Malgrati - Carlo Mantegazza - Stefano Milanesi - Maurizio Monteverdi - Sebastiano Pastore - Glauco Piatteletti - Daniele Riefoli - Alessandro Vallone - Giovanna Zampella

Stampa: Tipolito FE.ZA. Milano
Fotocomposizione: ERREGI Milano
Reg. del Trib. di Milano n. 519
del 10/11/84

Stampato in Italia